Speedup Potential of Climate-Based Daylight Modelling on GPUs

Nathaniel L Jones and Christoph F Reinhart
Massachusetts Institute of Technology, Cambridge, MA, USA

Abstract

Architects must consider an entire year's worth of solar positions and climate data to design buildings with adequate daylight and minimal glare. However, annual simulations are time-consuming and computationally expensive, which makes them difficult to integrate into iterative design processes. In this paper, we compare the performance of several RADIANCE-based dynamic daylighting simulation methods, DAYSIM and the three- and five-phase methods and, perhaps more importantly, the potential to speed them up using parallel implementations on graphic processors. Using a model of a generic office, we achieve speedups of ten times with DAYSIM and twenty-five times with the five-phase method. Parallel implementations of three- and five-phase methods provide better scaling to multi-GPU environments and more accurate results for complex fenestration systems than parallelized DAYSIM.

Introduction

Designing a naturally lit interior environment requires prediction and analysis of daylighting under all of sky conditions that occur in a building's climate. Annual daylighting simulations predict luminance distributions in a space for all daytime hours and are typically used to calculate climate-based daylight metrics (CBDMs). However, annual daylight simulations are quite time consuming, and as a result, it is difficult to run them in the quantities needed to explore a wide variety of design options. Parallel computation promises to make the results of annual daylight simulations available quickly, which will be especially beneficial early in the design process.

Making software programs run faster has long been an interest of the computer industry. In 1965, Gordon Moore put forth the idea now known as Moore's law, that the density of transistors on new integrated circuit chips, and by extension their computing power, doubles at a constant rate (Moore, 1965). Reliable speed increases were a direct result of this doubling until 2004, when thermodynamic and economic pressures caused chip manufacturers to change their strategy; as transistors continued to shrink, chips grew to accommodate multiple cores instead of faster clock speeds (Sutter, 2005). In the next few years, chip manufacturers expect to reach the limits of transistor density, below which quantum effects will make transistors unreliable (Waldrop, 2016). In this post-Moore era, continued software speedups need to come from parallelism and compiler optimization (Schardl, 2016). Graphics processing units (GPUs) follow a single-instruction multiple-thread (SIMT) computing model that allows individual cores to be made very small and is thus highly effective for running code in parallel (NVIDIA, 2016). Introducing a second layer of parallelism, large data processing jobs may be partitioned and distributed among multiple GPUs (Navarro, et al., 2014). In computational physics, multi-GPU environments can yield significant speedups (Thibault & Senocak, 2009; Jia, et al., 2012; Yokota, et al., 2012). We use parallelism to speed up CBDM calculations on one and two GPUs.

We have previously demonstrated parallelization of the ray-tracing kernel of DAYSIM at the primary ray level (Jones & Reinhart, 2015). While effective as a proof of concept, the solution did not scale well to large problems because of the memory requirements for storing a large number of daylight coefficient arrays on the GPU for parallel calculation. In this paper, we introduce an efficient way to parallelize annual daylighting simulations. We present a GPU-based version of *rcontrib*, the ray-tracing program used in the three-phase method (Ward, et al., 2011) and five-phase method (McNeil, 2013). Because *rcontrib* has a much smaller memory footprint that its DAYSIM counterpart, *rtrace_dc*, it scales better to large collections of sensors. Furthermore, our parallel version maintains the ability to analyze complex fenestration systems that the three- and five-phase methods were developed for, which we were unable to do in DAYSIM.

Background

Parallelism in Daylighting Simulation

In the building performance simulation community, daylighting simulation is typically performed using the RADIANCE suite of ray tracing software tools (Larson & Shakespeare, 1998). This suite includes the programs *rtrace* for point sensor simulation, *rpict* for image rendering (mimicking a high-dynamic range camera, which is an advanced sensor), *rvu* for model visualization, and *rcontrib* for dynamic daylight simulations. The related DAYSIM suite performs

annual daylighting calculations (Reinhart & Walkenhorst, 2001). While both sets of programs are accessible through command-line environments, a number of computer aided design tools and third-party plug-ins provide graphical interfaces to RADIANCE and DAYSIM.

RADIANCE and DAYSIM use light-backward ray tracing, which traces rays from sensor locations through multiple-bounce paths until either reaching a light source or being extinguished (Whitted, 1980). When multiple sensors are considered, we can trace rays from each independently and in parallel. The exception occurs when using irradiance caching, which applies an ambient value calculated for one ray to other rays that terminate in its vicinity. Various solutions have been proposed to parallelize irradiance caching, with varying degrees of success (Křivánek & Gautron, 2009; Wang, et al., 2009; Frolov, et al., 2013; Jones & Reinhart, 2016a).

Researchers have made a number of attempts to parallelize point-in-time RADIANCE simulations. RADIANCE itself allows limited parallelization by running multiple instances simultaneously with some sharing of data controlled by file locks. More sophisticated parallelization approaches using Message-Passing Interface (MPI) (Koholka, et al., 1999; Debattista, et al., 2006) and wait-free synchronization (Dubla, et al., 2009) have not passed into common use. Zuo et al. (2014) implemented the matrix multiplication portion of the three-phase method in parallel on GPUs, achieving a speedup of 800 times over previous methods for that step, but did not parallelize the ray tracing operations that account for most of the simulation time. In our own previous work on Accelerad, we have created GPU-accelerated parallel versions of *rtrace* and *rpict* (Jones & Reinhart, 2014a), *rtrace_dc* (Jones & Reinhart, 2015), and *rvu* (Jones & Reinhart, 2016b) with documented speedups up to 44 times faster than RADIANCE (Jones & Reinhart, 2017). We now turn our attention to the final ray-tracing program, *rcontrib*, to determine whether its parallelization can result in easier access to daylighting metrics.

Calculating Daylighting Metrics

Daylighting describes the use of natural light to illuminate building interiors. The amount of daylight present in a building at any given time depends on the sun's position and current weather conditions. Climate-based daylighting metrics (CBDMs) such as spatial daylight autonomy and annual sunlight exposure describe daylighting over a space's annual occupied hours (IESNA, 2012). These metrics are now integrated into compliance paths for both the LEED (USGBC, 2013) and WELL (International WELL Building Institute, 2016) green building standards. Spatial daylight autonomy describes the fraction of occupied space that receives at least 300 lux for at least 50% of occupied hours and is abbreviated $sDA_{300,50\%}$. Annual sunlight exposure is the fraction of occupied space that receives at least 1000 lux (and can therefore be assumed to be in direct sunlight) during at least 250 occupied hours and is abbreviated $ASE_{1000,250}$. Designers should attempt to maximize $sDA_{300,50\%}$ and minimize $ASE_{1000,250}$ to provide adequate natural illumination without overheating.

If we assume that a typical office is occupied for eight hours each day, or 2920 hours per year, then a naïve approach to calculating CBDMs would be to run 2920 point-in-time daylighting simulations over a grid of sensors with RADIANCE's *rtrace* program, once for each hour. On each iteration, *rtrace* would trace rays from each sensor through the scene along multiple-bounce paths until reaching the sun and sky and sampling the brightness of each. However, every simulation would trace the same ray paths and differ only by the brightness of the sky where the rays reach it. This approach results in considerable duplicated work and programmatic inefficiency.

DAYSIM and the three- and five-phase methods all avoid duplicating work by using the ray-tracing step to calculate matrix entries. The matrix (or product of matrices) is a transformation function between source radiance and sensor irradiance values. When multiplied by a vector containing a given sky condition, it produces an array of sensor values. Although the details of the methods differ, as we will describe later, both DAYSIM's *rtrace_dc* and the three- and five-phase methods' *rcontrib* are simple modifications of the original *rtrace* algorithm.

Validation

A significant number of quantitative studies support RADIANCE as an accurate simulation tool for daylit buildings. In one study, *rtrace* produced the most accurate results out of four validated rendering packages when compared to field measurements of a daylit interior, although still off by up to 40% (Ubbelohde & Humann, 1998). In a study of daylight control options, all *rtrace* relative errors were under 20% (Ng, et al., 2001). Because our CBDM calculation methods derive from *rtrace*, we expect similar performance from them.

This expectation is borne out through a number of studies of DAYSIM. Several studies comparing daylit interiors to DAYSIM predictions found relative mean bias error (MBE_{rel}) under 20% and relative root mean square error ($RMSE_{rel}$) under 32% (Reinhart & Walkenhorst, 2001; Reinhart & Andersen, 2006). DAYSIM gave comparable results to 3ds Max in a study of one building interior under a number of sky conditions (Reinhart & Breton, 2009) but offered superior results at four other geographic locations (Bellia, et al., 2015).

Validation of the three- and five-phase methods shows similar accuracy. The three-phase method produced close agreement with theoretical flux values for venetian blinds (Ward, et al., 2011) and had MBE_{rel} less than 13% and $RMSE_{rel}$ less than 23% for a light-redirecting component (McNeil & Lee, 2013). In a

study of four classrooms, the three- and five-phase methods gave similar $sDA_{300,50\%}$ results to each other and to DAYSIM, although more variance occurred in $ASE_{1000,250}$ (Brembilla, 2016). Images created with the three- and five-phase methods had similar appearance to conventional RADIANCE *rpict* images and produced similar image-based predictions of daylight glare probability (Inanici & Hashemloo, 2017).

Implementation

DAYSIM

In DAYSIM, the matrix entries are daylight coefficients. Each daylight coefficient represents the contribution of a light source to a sensor, such that the total illuminance at that point is the sum of all direct and diffuse daylight coefficients multiplied by the respective luminance values of their sources at a particular point in time (Tregenza & Waters, 1983). DAYSIM calculates direct and diffuse daylight coefficients separately (with *rtrace_dc*), creating two matrices and then concatenating them (with *gen_dc*). For the diffuse matrix D_{dif}, DAYSIM uses 148 sources corresponding to the 145 Tregenza sky divisions (Tregenza, 1987) and three ring-shaped ground patches. For the direct matrix D_{dir}, DAYSIM creates directional sources spaced uniformly along the solar paths of the chosen latitude (Reinhart & Walkenhorst, 2001). The number of direct daylight coefficients varies by latitude; 63 are needed at the authors' location. The final step (carried out by *ds_illum*) is to calculate the irradiance matrix I listing the irradiance at each sensor for each time of the year as follows:

$$I = D_{dir}S_{sun} + D_{dif}S_{sky} = \left(D_{dir}\middle|D_{dif}\right)\left(S_{sun}\middle|S_{sky}\right) \quad (1)$$

where the matrices S_{sun} and S_{sky} list the radiance of each sun position and sky patch, respectively, at each hour of the year. In practice, the values reported from DAYSIM have units of illuminance (lux) rather than irradiance (W/m^2), which is achieved by multiplying I by 179 lm/W (Inanici, et al., 2015).

Three- and Five-Phase Methods

The three-phase method may be understood as an evolution from DAYSIM. It differs in two key ways. First, the brightness of the sun is added to the sky dome to create a single sky matrix S (calculated by *gendaymtx*). This eliminates the need for separate direct and diffuse ray tracing passes but also removes hard shadows. Second, the daylight coefficient matrix is replaced with a product of three matrices (calculated by *rcontrib*). These are the daylight matrix D, relating light that reaches windows to its sources in the sky, the transmission matrix T, which is a bidirectional scattering distribution function (BSDF) that describes light passing through a window or complex fenestration system in terms of light incident on that surface, and the view matrix V, relating light leaving a window to the light arriving at sensors. The entries in each matrix are no longer daylight coefficients, since they do not describe the complete source-to-sensor relationship, so instead we call them contribution coefficients. The irradiance matrix is calculated (by *dctimestep*) as:

$$I = VTDS \quad (2)$$

Unlike *rtrace_dc*, *rcontrib* performs separate calculations in the red, green, and blue channels and interleaves them in the matrices. We convert the results to illuminance as follows:

$$L = 179 \times (0.2651r + 0.670g + 0.065b) \quad (3)$$

where L is illuminance in lux and r, g, and b are the red, green, and blue irradiance values in W/m^2 (Inanici, et al., 2015).

The five-phase method extends the three-phase method by separating the direct irradiance calculation. This makes it possible to render hard shadows and otherwise brings the method in line with the standard daylight coefficient model proposed by Bourgeois, et al. (2008). After running a normal three-phase method simulation, the next step is to isolate and remove the direct contribution from the previously calculated result. This means repeating the calculation of D and V with no light bounces (using a non-reflective, black version of the model) to determine how much light must be removed. These direct-only matrices D_d and V_d, are used together with a direct sun-only sky matrix S_{ds}. Finally, the direct sun component is added back in using a fine grid of suns centered in Reinhart sky patches, which are subdivisions of Tregenza sky patches (Bourgeois, et al., 2008). The use of the Reinhart sky patches stored in S_{sun} allows simulation results to be reused for multiple building orientations or geographic locations, although this flexibility is more useful in academic study than in practice. The calculation again uses a non-reflective, black model, with the exception of windows, which retain their transparency, and the calculation produces actual daylight coefficients in C_{ds} instead of contribution coefficients because BSDFs are not relevant to direct ray paths. The entire five-phase method is then:

$$I = VTDS - V_dTD_dS_{ds} + C_{ds}S_{sun} \quad (4)$$

Ray Payloads

Before we describe our modifications to parallelize *rcontrib* for the GPU, it is useful to cover some core concepts of ray tracing. Each ray contains both geometric information (origin and direction) and a payload. Usually, the payload is a color or radiance value that is calculated when the ray intersects a surface. Unless the surface emits its own light, this calculation generally requires tracing new reflected rays. After tracing the entire tree of rays, the payload returned by the primary ray at the tree's root becomes the value of its associated sensor or pixel.

Both DAYSIM's *rtrace_dc* and RADIANCE's *rcontrib* augment the ray payload. The former includes an array of 148 daylight coefficients in the payload for each

ray, which internal calculations treat like color channels. This array fills 592 bytes, easily eclipsing the rest of the ray's data in size, and this substantially increases the memory use of *rtrace_dc* when computing many rays in parallel. In contrast, *rcontrib* adds three double-precision ray coefficients to the ray payload, one each for the red, green, and blue channels. The cost per ray for the additional payload is only 24 bytes. Rather than accumulating value, as a color or daylight coefficient payload would, ray coefficients represent the weighting factor of each color channel in calculating that channel's value for the parent ray. When a ray hits a surface or source with a material of interest specified by the user (usually a light source), *rcontrib* adds the cumulative product of the ray coefficients in the current tree to a contribution coefficient for that material. At the conclusion of ray tracing, the program outputs the coefficients rather than the radiance values. Any lighting condition produced by a set of sources (or sky patches) is a linear combination of the contribution coefficients.

Because *rcontrib* calculates coefficients at leaf nodes of the ray tree rather than at the root, as DAYSIM does, it is incompatible with irradiance caching. However, irradiance caching artificially increases the sampling importance of diffuse rays far from the root. Without it, the minimum ray weight would need to be set extremely low to simulate diffuse lighting accurately, which would severely increase simulation times. To avoid this, the three- and five-phase methods advise the use of Russian roulette to terminate ray tracing; ray paths terminate at random beyond a certain depth in the tree, and the remaining rays receive accordingly higher weights (Pharr & Humphreys, 2010). The *rcontrib* program enables Russian roulette by default.

Our Modifications

We make two changes to the *rcontrib* process in order to parallelize it for the GPU. First, our ray coefficients store weights relative to the primary ray instead of the immediate parent ray. This prevents the program from having to access pointers to many rays in each ray intersection calculation, which puts less strain on the limited number of registers available to each GPU thread and avoids spills into global memory. In our analysis, we tested the effect of storing single-precision ray coefficients, which may be computed faster on GPUs, versus double-precision ray coefficients, which are more resistant to numerical error propagation.

Second, we store one set of contribution coefficients per root ray in global GPU memory. We assign each working thread on the GPU its own array of m contribution coefficients (one per material or per sky patch) in global memory that it populates independently of the other GPU threads. For a model with n sensors, the memory size is $n \times z$ bytes, where the depth of bytes per sensor is:

$$z = CC \times m \quad (5)$$

where CC is the size of a contribution coefficient RGB triplet in bytes. The array must reside in global memory because of its size, but it is accessed infrequently and therefore does not slow program execution significantly. We contrast this to DAYSIM, which reads and writes to the daylight coefficient array at every ray intersection.

Results

In order to compare the performance and speedup potential through GPU parallelism of DAYSIM with those of the three- and five-phase methods, we performed annual daylight analysis on a set of four models with all three methods. The first models used were small, medium, and large versions of the south-facing reference office (Reinhart, et al., 2013). The office interior measures 3.6 by 8.2 meters and contains a grid of 1425 irradiance sensors at 0.15-meter spacing located one meter above the floor (Figure 1). The small version of the model consisted of a single reference office at the authors' latitude, while the medium and large models consisted of two and ten side-by-side copies of the office yielding 2850 and 14250 sensors, respectively. We define model size in terms of the number of sensors involved in the analysis because this quantity directly affects simulation time. The modularity of our models also forces DAYSIM's irradiance cache to grow proportionally with model size rather than treating it as a separate variable. The fourth model was the same as the first with the addition of exterior blinds.

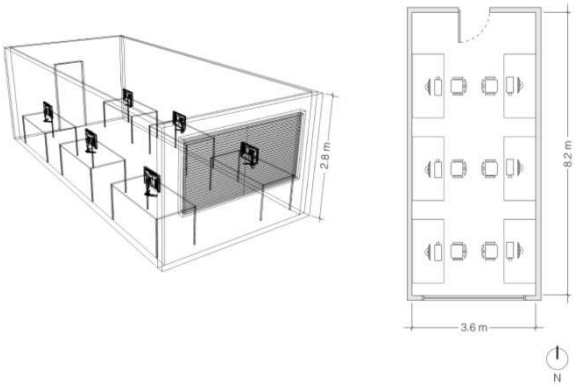

Figure 1: The reference office, shown in perspective and plan views, contains six workstations with a south-facing window (Reinhart, et al., 2013).

We chose simulation settings recommended by other sources. Because DAYSIM's irradiance caching and the three- and five-phase methods' Russian roulette are fundamentally different approaches for diffuse calculations, each recommends different settings. We ran DAYSIM using the high-accuracy settings recommended by DIVA-for-Rhino (Jakubiec & Reinhart, 2011) version 4 with some modifications specific to the size of our model. We ran the three- and five-phase methods with the settings recommended by

Andrew McNeil (2010; 2013). The GPU-based versions used the same settings as their serial counterparts, except for the addition of the *–ac* parameter to control irradiance cache size (Jones & Reinhart, 2014b). Although the default size 4096 sufficed for the small and medium models, a proportionately larger value was necessary for the large model. Table 1 lists the settings we used.

Table 1: Default simulation parameters

Parameter	DAYSIM	3-/5-PM
Ambient accuracy (*–aa*)	0.05	0
Ambient bounces (*–ab*)	8	8
Ambient divisions (*–ad*)	4096	50000
Ambient resolution (*–ar*)	300	256
Ambient super-samples (*–as*)	20	0
Direct jitter (*–dj*)	0	0.9
Direct relays (*–dr*)	2	3
Direct sampling (*–ds*)	0.2	0.2
Max. ray reflections (*–lr*)	6	-10
Minimum ray weight (*–lw*)	0.001	0.00002
Specular sampling (*–ss*)	1	1
Specular threshold (*–st*)	0.15	0.15
Irradiance cache size (*–ac*) *GPU only*	4096/16384	N/A

The code used in all of our simulations was based on RADIANCE release 5.0.a.12 and compiled for Windows. This allowed a fairer comparison between methods, but it required a custom compilation of DAYSIM since the publicly available version at the time of writing was based on an older RADIANCE release. To create the parallel versions of both programs, we replaced RADIANCE's own ray tracing code with calls to the OptiX™ GPU ray tracing library from NVIDIA® (Parker, et al., 2010), similar to our method for creating other Accelerad programs.

We ran simulations on two machines. The serial implementations ran on a workstation with a 2.60 GHz Intel® Xeon® E5-2604 processor. The parallel implementations ran on a workstation with a 2.27 GHz Intel® Xeon® E5520 processor and two NVIDIA® Tesla® K40 graphics accelerators with 2880 CUDA® cores each. Using the slower workstation for the parallel simulations was necessary because the faster workstation lacked sufficient power supply for the graphics accelerator cards we used.

Daylight Metrics

The six simulation methods generally produced similar results for $sDA_{300,50\%}$ and $ASE_{1000,250}$ in the four models (Figure 2). For the small, medium, and large reference office models, which should yield identical CBDM results, all simulations predicted $sDA_{300,50\%}$ within half a percent of 50%, with the exception of the parallel DAYSIM simulation. This is consistent with earlier observations in which parallelized *rtrace_dc* tended to predict lower $sDA_{300,50\%}$ than the serial version (Jones & Reinhart, 2015). The discrepancy grows with model size.

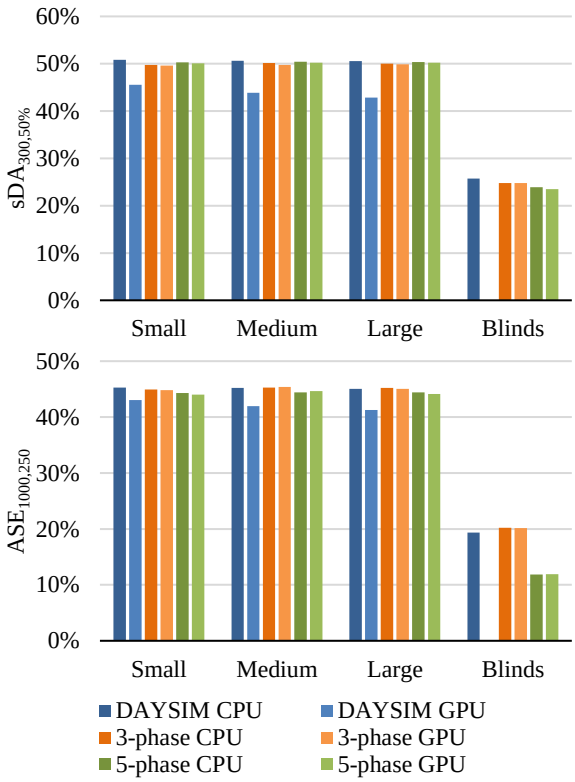

Figure 2: Spatial daylight autonomy ($sDA_{300,50\%}$) and annual sunlight exposure ($ASE_{1000,250}$) for the models.

For the model with blinds, slightly more variation existed between the simulation results. Serial DAYSIM predicted an $sDA_{300,50\%}$ of 26%, while the three- and five-phase methods predicted 24% and 25% respectively. These discrepancies are well within our error tolerance. However, the parallel DAYSIM implementation gave poor results. While the simulation did predict some light entering the room, the fixed size irradiance cache did not register enough diffuse light entering the space to meet the 300-lux threshold at any sensor.

We saw similar results for $ASE_{1000,250}$. The parallel DAYSIM results again diverged from the other simulation methods, which predicted an $ASE_{1000,250}$ of 45% for the unshaded models. A significant discrepancy arose between the five-phase method and the other simulation methods for the model with blinds. The serial DAYSIM simulation and both three-phase method simulations yielded similar $ASE_{1000,250}$ predictions around 20%, while the five-phase method gave a lower prediction of 12%. The lower value occurred because the five-phase method's direct sun-only term did not account for interreflection within the blinds and was therefore much smaller than the subtracted direct-only component that incorporated the blinds' BSDF.

Figure 3 shows example results from each of the six simulations. The illuminance snapshots show lighting conditions under a single sky condition. The relatively small number of sun positions considered by DAYSIM results in several shadows cast by the

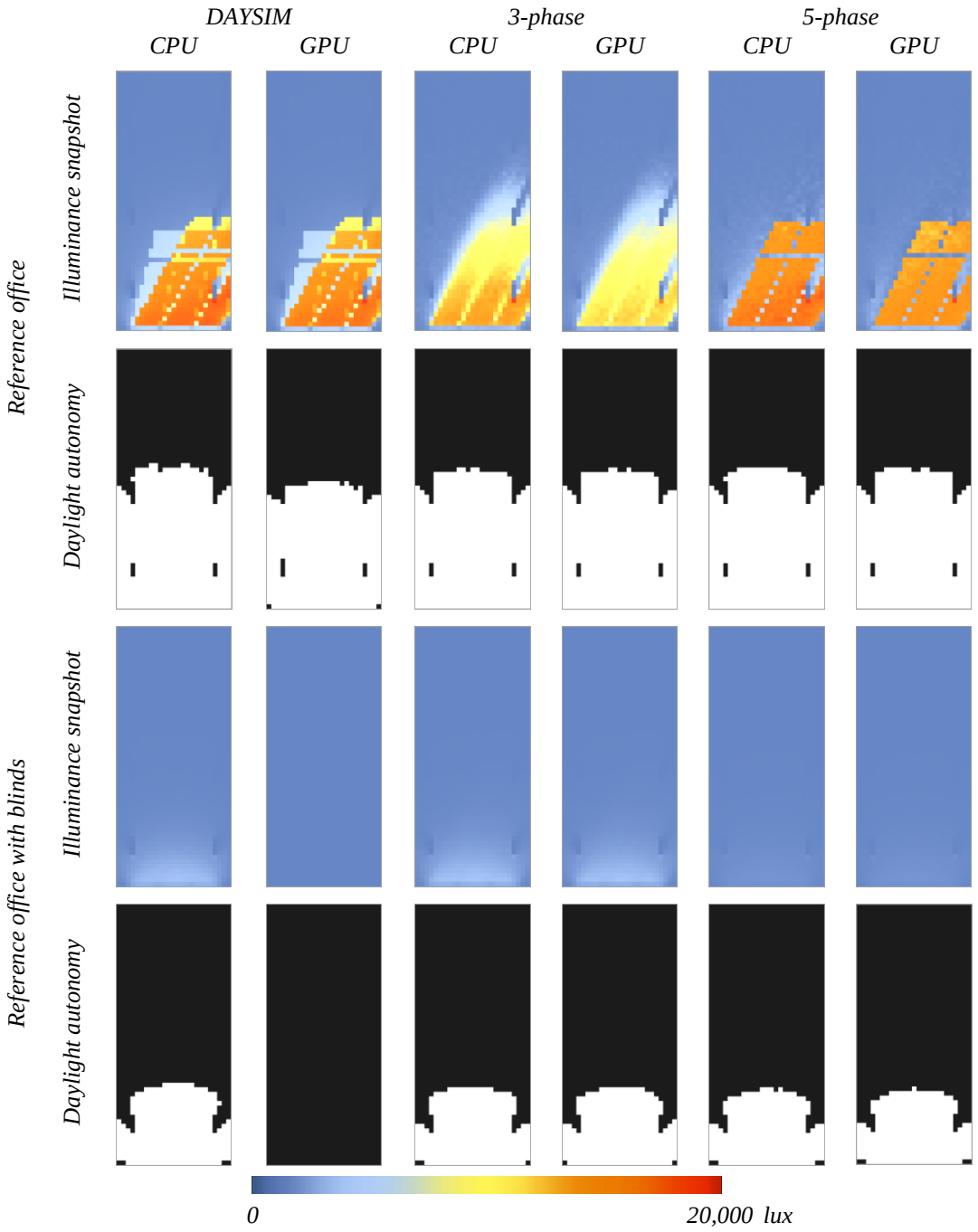

Figure 3: Simulation results of the reference office without and with blinds. Illuminance snapshots show lighting conditions at 2pm on January 1st. Daylight autonomy images show regions that achieve the IESNA standard LM-83 target of 300 lux for at least 50% of occupied hours in white.

window. The three-phase method did not produce well-defined shadows and instead gave the pattern of light entering through the window a smudged appearance. The five-phase method considers a large number of sun positions, so it was able to produce a single, hard-edged shadow. The parallel three- and five-phase simulations produced lower maximum brightness values than their respective serial versions, but as the maximum values were significantly higher than 1000 lux, these differences were not apparent in either $sDA_{300,50\%}$ or $ASE_{1000,250}$ results.

Speedup

We ran each simulation in serial on the Intel® Xeon® E5-2604 workstation and in parallel on one and two Tesla® K40 accelerators. Our concern here lies with the time taken to run *rtrace_dc* or *rcontrib* in each case. Although the matrix algebra performed by *ds_illum* and *dctimestep* also added to the total simulation time, its contribution was relatively minor and in any case was unchanged by parallelizing the ray-tracing portion of the simulation. For each simulation type, we report the mean runtime from

Table 2: Matrix calculation times by rtrace_dc *and* rcontrib *in minutes.*

Model	Processor	DAYSIM		3-/5-phase			5-phase		
		D_{dir}	D_{dif}	D	T	V	D_d	V_d	C_{ds}
Small	CPU	39.2	22.9	1.6	0.1	10.3	0.0	2.1	600.3
	1x GPU	8.8	3.9	0.2	0.1	2.3	0.1	0.5	21.1
	2x GPU	5.6	2.8	0.7	0.2	3.3	0.6	0.7	45.2
Medium	CPU	78.0	45.5	1.6	0.1	20.9	0.0	4.1	1219.7
	1x GPU	12.0	6.3	0.2	0.1	6.2	0.1	1.2	85.6
	2x GPU	7.8	4.2	0.7	0.2	6.2	0.6	1.3	73.2
Large	CPU	331.5	177.8	1.7	0.1	111.2	0.0	22.6	6031.0
	1x GPU	142.4	117.0	0.2	0.1	32.5	0.2	5.8	374.9
	2x GPU	84.6	68.2	0.7	0.2	21.3	0.6	4.4	323.9
Blinds	CPU	96.1	42.9	1.6	21.8	10.1	0.0	2.1	623.5
	1x GPU	9.1	4.1	0.2	0.8	2.3	0.1	0.5	21.6
	2x GPU	5.8	2.9	0.7	0.7	3.2	0.6	0.7	47.8

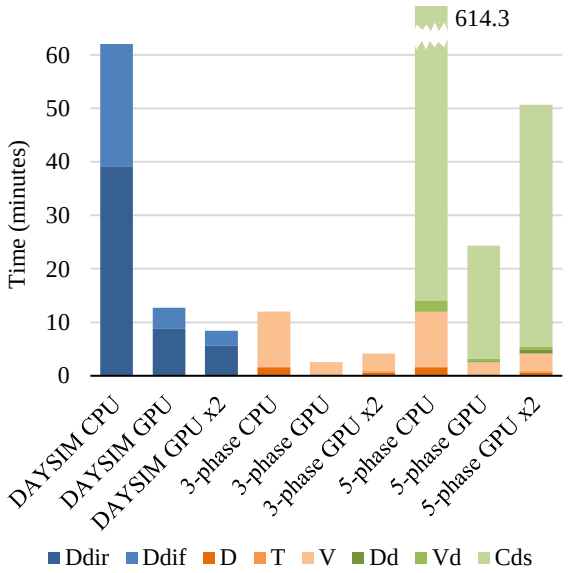

Figure 4: Cumulative matrix calculation times by rtrace_dc *and* rcontrib *for the small model.*

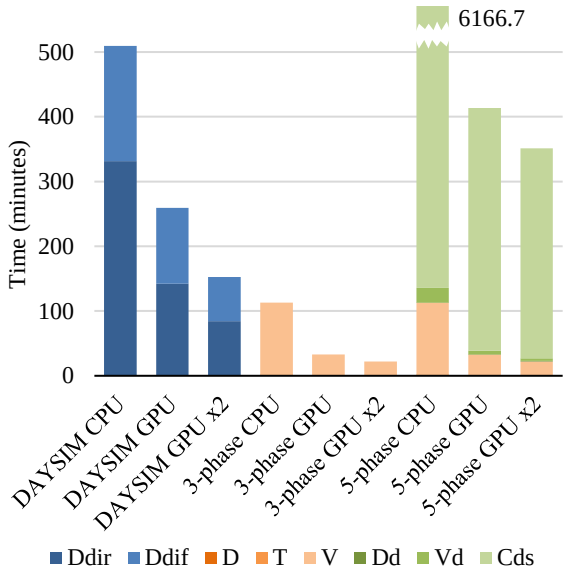

Figure 5: Cumulative matrix calculation times by rtrace_dc *and* rcontrib *for the large model.*

eight simulations. Table 2 breaks down the time taken to compute each matrix in each test, and Figure 4 and Figure 5 illustrate times for the small and large models, respectively.

In all cases, the three-phase method offered a speed advantage over DAYSIM, whether in parallel or not. This may come as a surprise given the higher-accuracy parameters recommended for the three-phase method and may indicate that Russian roulette offers more efficiency than irradiance caching in calculating diffuse lighting.

Calculation times for the daylight matrices D and D_d show little variance. All of the models we tested had identical views to the sky from each window, so calculation times did not vary with model size. The GPU calculation times can be reduced for these matrices by storing the contribution coefficients with single precision instead of double precision. The calculation was also faster using a single GPU rather than both together. These facts indicate that loading programs and copying memory to and from the GPUs took up more time than the calculations themselves. Indeed, calculation of D_d was fast enough on the CPU that parallelism had no benefit at all.

Calculation times for the view matrices V and V_d and the direct sun matrix C_{ds} did benefit from parallelism. This was particularly true for V, which involved a large number of ray bounces, and C_{ds}, which cast a large number of shadow rays because of the number of sun positions. For the smaller models, calculations ran faster on a single GPU than when using both (Figure 4). The smaller models had only 1425 and 2850 sensors, not enough to justify the use of a second GPU. (Each GPU had 2880 cores, though we cannot assume a one-to-one assignment of sensor to core by the graphics driver.) In contrast, the large model had enough sensors that splitting work between GPUs did result in faster simulations (Figure 5).

Calculation times for the transmission matrix *T* benefit from parallelism, but only for complex fenestration systems. In the model with blinds, the BSDF calculation ran twenty-six times faster on a single GPU and thirty-one times faster on both GPUs. In the models without shading devices, there was no advantage to using the GPU. The timings reported in Table 2 do not include the run times of *genBSDF* and *rfluxmtx*, which wrap the *rcontrib* BSDF calculation, but the overhead of those programs is minimal.

Figure 6 summarizes the speedup factor achieved by each test. The best improvement was a twenty-five-fold speedup on the small model with the five-phase method. More noteworthy, however, are the trends that emerged in scalability with model size. Parallel DAYSIM performed well for small models, but the speedup factor decreased for larger models. This is mainly due to the necessity of using a larger irradiance cache with larger models. On the other hand, the three- and five-phase methods did not show lessening speedups between the medium and large models, and in fact showed improvement when using multiple GPUs.

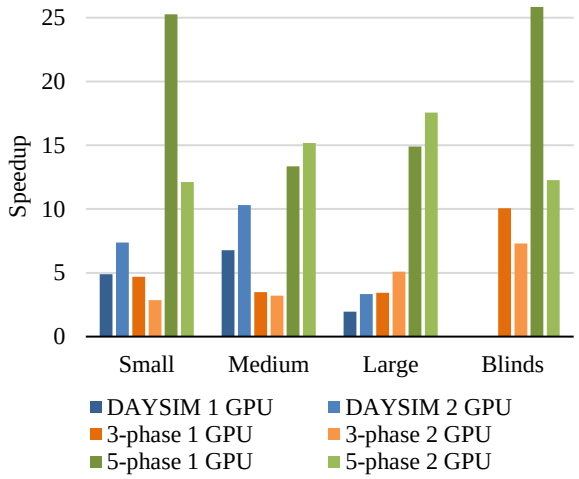

Figure 6: Speedup factors for each method using one or two Tesla K40 accelerators. DAYSIM speedups for the blinds model are not shown because the results are not useable.

Conclusions

This study demonstrates the potential of GPU computation to speed up CBDM simulations. We tested three simulation methods, DAYSIM and the three- and five-phase methods, in serial and parallel. The parallel version of *rcontrib* used by the three- and five-phase methods produced more reliable results than the parallel version of *rtrace_dc* used by DAYSIM, and its speedup scaled better with model size.

Ultimately, the choice of what method to use to calculate CBDMs depends on the situation. For small models without complex fenestration systems, all of the methods will produce useful results. Designers seeking only CBDMs and not images should use the three-phase method to get the fastest results, which can be available within minutes using parallelism on a single GPU. Where the definition of accurate hard shadows is important, designers should use the five-phase method, but parallel calculation may be necessary in order to keep the simulation time competitive with other methods.

For large models with thousands of sensors or more, multi-GPU environments provide better scaling and improved speedup. However, DAYSIM does not scale as well to multi-GPU environments, mainly because of its reliance on irradiance caching. Similarly, DAYSIM is not a reliable platform for analysing complex fenestration systems, especially in parallel. Designers should therefore use the GPU-based three- and five-phase methods when working with large models.

Given our increased reliance on CBDMs for building assessment and rating systems, we expect typical model sizes and complexities to increase in the future. This necessitates faster simulation tools to meet the need for more frequent and more complex simulations. We can no longer rely on Moore's law to make serial simulations faster; instead, new generations of parallel computer architectures provide the best hope to meet simulation needs. Our research points out the need for computer hardware with increased core counts and reduced memory transfer overhead. Putting these efficiencies to use will allow architects and lighting designers to calculate CBDMs earlier and more frequently in the design process, leading to better daylighting performance in buildings.

Acknowledgement

The Tesla K40 accelerators used for this research were donated by the NVIDIA Corporation. The authors thank Eleonora Brembilla for her contributions to validating the three- and five-phase methods.

References

Bellia, L., Pedace, A. & Fragliasso, F., 2015. The impact of the software's choice on dynamic daylight simulations' results: A comparison between Daysim and 3ds Max Design®. *Solar Energy*, Volume 122, pp. 249-263.

Bourgeois, D., Reinhart, C. & Ward, G., 2008. Standard daylight coefficient model for dynamic daylighting simulations. *Building Research & Information*, 36(1), pp. 68-82.

Brembilla, E., 2016. Applicability of climate-based daylight modelling. In: *Young Lighter of the Year Competition 2016*. Society of Light and Lighting, UK: https://dspace.lboro.ac.uk/2134/23273.

Debattista, K., Santos, L. P. & Chalmers, A., 2006. Accelerating the irradiance cache through parallel component-based rendering. *Proceedings of the 6th Eurographics conference on Parallel Graphics and Visualization*, pp. 27-35.

Dubla, P., Debattista, K., Santos, L. P. & Chalmers, A., 2009. Wait-free shared-memory irradiance cache. *Proceedings of the 9th Eurographics Symposium on Parallel Graphics and Visualization*, pp. 57-64.

Frolov, V., Vostryakov, K., Kharlamov, A. & Galaktionov, V., 2013. Implementing irradiance cache in a GPU photorealistic renderer. In: M. L. Gavrilova, C. K. Tan & A. Konushin, eds. *Transactions on Computational Science XIX*. Berlin: Springer, pp. 17-32.

IESNA Daylighting Metrics Committee, 2012. *Lighitng Measurement #83, Spatial Daylight Autonomy (sDA) and Annual Sunlight Exposure (ASE)*, New York: IESNA Lighting Measurement.

Inanici, M., Brennan, M. & Clark, E., 2015. Spectral daylighting simulations: Computing circadian light. *Proceedings of BS2015: 14th Conference of International Building Performance Simulation Association, Hyderabad, India, Dec. 7-9, 2015*, pp. 1245-1252.

Inanici, M. & Hashemloo, A., 2017. An investigation of the daylighting simulation techniques and sky modeling practices for occupant centric evaluations. *Building and Environment*, Volume 113, pp. 220-231.

International WELL Building Institute, 2016. *The WELL Building Standard® v1*, New York: Delos Living LLC.

Jakubiec, J. A. & Reinhart, C. F., 2011. DIVA 2.0: Integrating daylight and thermal simulations using Rhinoceros 3D and EnergyPlus. *Proceedings of Building Simulation 2011: 12th Conference of International Building Performance Simulation Association, Sydney, 14-16 November*, pp. 2202-2209.

Jia, Y., Luszczek, P. & Dongarra, J., 2012. Multi-GPU implementation of LU factorization. *Procedia Computer Science*, Volume 9, pp. 106-115.

Jones, N. L. & Reinhart, C. F., 2014a. Physically based global illumination calculation using graphics hardware. *Proceedings of eSim 2014: The Canadian Conference on Building Simulation*, pp. 474-487.

Jones, N. L. & Reinhart, C. F., 2014b. Irradiance caching for global illumination calculation on graphics hardware. *2014 ASHRAE/IBPSA-USA Building Simulation Conference, Atlanta, GA, September 10-12*, pp. 111-120.

Jones, N. L. & Reinhart, C. F., 2015. Fast daylight coefficient calculation using graphics hardware. *Proceedings of BS2015: 14th International Conference of the International Building Performance Simulation Association, Hyderabad, India, Dec. 7-9, 2015*, pp. 1237-1244.

Jones, N. L. & Reinhart, C. F., 2016a. Parallel multiple-bounce irradiance caching. *Computer Graphics Forum*, 35(4), pp. 57-66.

Jones, N. L. & Reinhart, C. F., 2016b. Real-Time Visual Comfort Feedback for Architectural Design. *PLEA 2016 Los Angeles – 32nd International Conference on Passive and Low Energy Architecture*, pp. 659-664.

Jones, N. L. & Reinhart, C. F., 2017. Experimental validation of ray tracing as a means of image-based visual discomfort prediction. *Building and Environment*, Volume 113, pp. 131-150.

Koholka, R., Mayer, H. & Goller, A., 1999. MPI-parallelized Radiance on SGI CoW and SMP. *Proceedings of the 4th International ACPC Conference Including Special Tracks on Parallel Numerics and Parallel Computing in Image Processing, Video Processing, and Multimedia: Parallel Computation*, pp. 549-558.

Křivánek, J. & Gautron, P., 2009. Practical global illumination with irradiance caching. *Synthesis Lectures on Computer Graphics and Animation*, 4(1), pp. 1-148.

Larson, G. W. & Shakespeare, R., 1998. *Rendering with Radiance: The Art and Science of Lighting Visualization.* San Francisco: Morgan Kaufmann Publishers, Inc.

McNeil, A., 2010. *The Three-Phase Method for Simulating Complex Fenestration with Radiance*.

McNeil, A., 2013. *The Five-Phase Method for Simulating Complex Fenestration with Radiance*.

McNeil, A. & Lee, E., 2013. A validation of the Radiance three-phase simulation method for modelling annual daylight performance of optically complex fenestration systems. *Journal of Building Performance Simulation*, 6(1), pp. 24-37.

Moore, G. E., 1965. Cramming more components onto integrated circuits. *Electronics*, 38(8), pp. 114-117.

Navarro, C. A., Hitschfeld-Kahler, N. & Mateu, L., 2014. A survey on parallel computing and its applications in data-parallel problems using GPU architectures. *Communications in Computational Physics*, 15(2), pp. 285-329.

Ng, E. Y.-Y., Poh, L. K., Wei, W. & Nagakura, T., 2001. Advanced lighting simulation in architectural design in the tropics. *Automation in Construction*, 10(3), pp. 365-379.

NVIDIA, 2016. *CUDA C Programming Guide*, PG-02829-001_v8.0, September 2016.

Parker, S. G. et al., 2010. OptiX: A general purpose ray tracing engine. *ACM Transactions on Graphics – Proceedings of ACM SIGGRAPH 2010*, 29(4).

Pharr, M. & Humphreys, G., 2010. *Physically Based Rendering: From Theory to Implementation.* 2nd Edition ed. Burlington: Morgan Kaufmann.

Reinhart, C. & Breton, P.-F., 2009. Experimental validation of Autodesk® 3ds Max® Design 2009 and Daysim 3.0. *LEUKOS: The Journal of the Illuminating Engineering Society of North America*, 6(1), pp. 7-35.

Reinhart, C. F. & Andersen, M., 2006. Development and validation of a radiance model for a translucent panel. *Energy and Buildings*, 38(7), pp. 890-904.

Reinhart, C. F., Jakubiec, J. A. & Ibarra, D., 2013. Definition of a reference office for standardized evaluations of dynamic façade and lighting technologies. *Proceedings of BS2013: 13th Conference of International Building Performance Simulation Association, Chambéry, France, August 26-28*, pp. 3645-3652.

Reinhart, C. F. & Walkenhorst, O., 2001. Validation of dynamic RADIANCE-based daylight simulations for a test office with external blinds. *Energy and Buildings*, Volume 33, pp. 683-697.

Schardl, T. B., 2016. *Performance engineering of multicore software: Developing a science of fast code for the post-Moore era*, PhD thesis: Massachusetts Institute of Technology.

Sutter, H., 2005. A fundamental turn toward concurrency in software. *Dr. Dobb's Journal*, 30(3), pp. 16-22.

Thibault, J. & Senocak, I., 2009. CUDA implementation of a Navier-Stokes solver on multi-GPU desktop platforms for incompressible flows. *47th AIAA Aerospace Sciences Meeting Including The New Horizons Forum and Aerospace Exposition, 5 – 8 January 2009, Orlando, Florida*, pp. 1-15.

Tregenza, P., 1987. Subdivision of the sky hemisphere for luminance measurements. *Lighting Research and Technology*, Volume 19, pp. 13-14.

Tregenza, P. & Waters, I., 1983. Daylight coefficients. *Lighting Research and Technology*, 15(2), pp. 65-71.

Ubbelohde, M. S. & Humann, C., 1998. Comparative evaluation of four daylighting software programs. In: *1998 ACEEE Summer Study on Energy Efficiency in Buildings, Pacific Grove, CA (US), 08/23/1998–08/28/1998.* Washington: American Council for an Energy-Efficient Economy, pp. 3.325-3.340.

US Green Building Council (USGBC), 2013. *LEED Reference Guide for Building Design and Construction, LEED V4*, Washington DC: USGBC.

Waldrop, M. M., 2016. The chips are down for Moore's law. *Nature News*, Volume 530, pp. 144-147.

Wang, R., Zhou, K., Pan, M. & Bao, H., 2009. An efficient GPU-based approach for interactive global illumination. *ACM Transactions on Graphics – Proceedings of ACM SIGGRAPH 2009*, 28(3).

Ward, G. et al., 2011. Simulating the daylight performance of complex fenestration systems using bidirectional scattering distribution functions within Radiance. *LEUKOS: The Journal of the Illuminating Engineering Society of North America*, 7(4), pp. 241-261.

Whitted, T., 1980. An improved illumination model for shaded display. *Communications of the ACM*, 23(6), pp. 343-349.

Yokota, R., Barba, L., Narumi, T. & Yasuoka, K., 2012. Scaling fast multipole methods up to 4000 GPUs. *Proceedings of the ATIP/A*CRC Workshop on Accelerator Technologies for High-Performance Computing: Does Asia Lead the Way?*, pp. 9:1-9:6.

Zuo, W., McNeil, A., Wetter, M. & Lee, E. S., 2014. Acceleration of the matrix multiplication of Radiance three phase daylighting simulations with parallel computing on heterogeneous hardware of personal computer. *Journal of Building Performance Simulation*, 7(2), pp. 152-163.